Neurocomputing, Volume 404, 3 September 2020, pp. 41-49
DOI: 10.1016/j.neucom.2020.04.107

Optimal Topology for Consensus using Genetic Algorithm

Sabyasachi Mondal[a,1,*], Antonios Tsourdos[b,2]

[a] *Cranfield, United Kingdom*
[b] *Cranfield, United Kingdom*

Abstract

In the Multi-Agent Systems (MAS), graph network topologies play a crucial role in building consensus among the connected agents. Consensus may be achieved on many network graphs using distributed control theory. However, the optimal network topology is not addressed in most of the literature, which is an important part of building stable consensus among networked agents. In this paper, the optimal topology is obtained irrespective of the agent dynamics by using two-dimensional Genetic Algorithm (GA), which is a new approach in this context. Simulation result for agents with first, and second-order linear dynamic is obtained. These results show that the proposed method achieves consensus using the optimal network topology satisfactorily.

Keywords: Distributed Control, Optimal Topology, Consensus, Two-dimensional Genetic Algorithm

1. Introduction

In recent time Multi-Agent Systems has been an active research area to deal with real-world problems. Practical problem scenarios are complex enough to handle with a single agent. Complex situations require Multi-Agent Systems

*Corresponding author
Email addresses: sabyasachi.mondal@cranfield.co.uk (Sabyasachi Mondal), a.tsourdos@cranfield.ac.uk (Antonios Tsourdos)
[1]Research Fellow, Aerospace Engineering, Cranfield University, MK430AL, UK
[2]Professor, Aerospace Engineering, Cranfield University, MK430AL, UK

which is a platform to execute complex tasks in a cooperative manner. Cooperation among the agents means to decide the actions for each agent considering the state information of the neighbouring agents obtained through a communication network. This interaction among the agents helps them to converge to a common value, which is known as consensus. In this process, a group of agents reaches an agreement to achieve a common goal. A considerable amount of work on a consensus problem has been reported in the literature. Some of them are mentioned in [1], [2], [3], [4], etc. Other examples of collective behaviour of cooperative platform are formation control [5, 6], Synchronization [7], [8], etc.

The interacting agents exchange information with its neighbour on a graph network to decide the strategy or control action for the consensus. The distributed control theory considers the interaction among the agents while selecting the control strategy. It essentially means the control strategy of an agent depends on how it is connected to its neighbour, and their dynamics. Distributed control protocols for different dynamical systems have been discussed in the literature. Consensus problem for the first-order system was studied in [9], [10] etc. The consensus was explained with the help of graph theory in [9] for Vicsek model [11] which proposed a system consisting of n agents and demonstrated by simulation that all agents asymptotically moved to one direction with the same speed. Consensus for second-order systems are studied in [12], [13], [14] etc. These works consider homogeneous linear systems in the multi-agent structure. There exist research works that consider first and second-order systems together in a multi-agent platform. Some of the works are mentioned in [15], [16], [17] etc. Also few works have been reported in [18], [19] which considers nonlinear nodal dynamics.

The information exchange with neighbours over communication the network plays a key role in building the consensus among the agents [20], [1]. The communication network can be analyzed using the graph theory [21]. The properties of graph theory have been proved to be very useful in analyzing the nature of connectivity among the agents. Some papers contain works related to communication topology. These works consider the switching topology, time delay in

communication, link failure, etc. In [22], consensus problems for dynamic agents with fixed and switching topologies are discussed. Three cases were discussed. The first one is directed networks with fixed topology; the second one is directed networks with switching topology, and the third one is undirected networks with communication time-delays and fixed topology. In [1], the problem of consensus in the Multi-Agent Systems on a dynamically changing topology is discussed while considering limited and unreliable communication. In [10], the consensus of MAS subjected to coupling delay and switching topology is analyzed. In [23], actuator and communication link fault in networked MAS is discussed. In [24] stochastic communication link failure in distributed leader-follower MAS is investigated. Data dropout for each independent communication link is analyzed using Bernoulli distribution.

In most of the papers, it is assumed that there exists a predefined communication topology and the optimal consensus is studied considering that specific topology, but there may exists a different network topology in which the agents can consume less control energy while using the same consensus control protocol. Obviously there exists one Optimal topology in which the agents spend minimum the minimum among all the possible network topologies. Our work is focused on finding this Optimal topology. The output of our work is the topology (more precisely an adjacency matrix) in which the agents should be connected such that they can achieve the consensus by spending minimum energy (using any consensus protocol). This problem has been studied in [25], [26], [27] where Linear Quadratic Regulator (LQR) was implemented to obtain a complete graph for the homogeneous system. The similar kind of work is done in [16], which considered the heterogeneous agents. The optimal topology obtained is a star graph which shows the existence of direct links between the leader and the followers. But this topology is restrictive since there is need of one dedicated connection between the leader and each follower. Moreover the weight of the edges are also fixed but they can change in practical situation, which may lead to change the cost as well.

In this paper, the optimal topology problem is addressed to find an optimal

communication topology using bio-inspired optimization technique like Genetic Algorithm (GA) which eliminates the need to solve the Algebraic Riccati Equation (ARE) used in LQR. Moreover, finite-time convergence is required to obtain fast consensus, which needs Riccati differential equations to be solved. The number of differential equation grows with the number of agents. In addition, this process relaxes the need of direct communication between the leader and followers (obtained using LQR). It is important to note that normal GA can't be used. Normal GA means GA with chromosome as one-dimensional array of bits. The solution of the optimal topology problem is an optimal adjacency matrix. Therefore, the chromosomes are two-dimensional matrix instead of a linear array.

The Contributions and significance of this work is summarized as follows:

- Introduction of Bio-inspired algorithm is the first attempt in consensus problem.
- The use of two-dimensional Genetic Algorithm in network topology optimization problem is a novel concept.
- The technique is applicable to all category of agents having Linear and Nonlinear dynamics. Moreover the proposed technique also relaxes the need of direct communication between the leader and each follower (discussed in a few papers mentioned above).
- The significance of the work is to design more energy efficient multi-agent platform (as per the control energy is concerned, which is limited) to execute complex operations. Minimization of the graph energy results in a reduction in number inter-agent communication links, i.e., the edge of the graph.

The two-dimensional GA and its use to obtain an optimal topology for consensus are discussed in the following sections.

2. Preliminaries

In this section, a brief discussion about the topics which are relevant to this work is provided. The relevant topics are the graph theory and consensus in the Multi-Agent Systems using distributed control. These topics are discussed in brief in the following section.

2.1. Graph Theory

A weighted directed graph can be described by $G = \{V, E\}$ where, $V = \{1, 2, \ldots, n\}$ represents a set of n vertices, $E = e_{ij} \in V \times V$ represents the set of edges. It can be mentioned that, in case of directed graph e_{ij} is the edge from j to i. The connectivity among the nodes or vertices is given in Adjacency matrix $A = [a_{ij}] \in R^{n \times n}$. The diagonal elements of adjacency matrix A are zero, i.e., $i \in V$, $a_{ii} = 0$. The off-diagonal elements, i.e., $\forall$ $i \neq j, e_{ij} \in E$, $a_{ij} > 0$ represents the weight associated to edge e_{ij}, while $a_{ij} = 0$ otherwise. $N_i = j : a_{ij} > 0$ denote the set of neighboring agent i. The degree matrix $D \in R^{n \times n} = diag\{\sum_{j \in N_1} a_{1j} \ldots \sum_{j \in N_n} a_{nj}\}$ and the Laplacian matrix is $L = D - A$. An example graph is shown in fig. 1 and the matrices associated to it are given as follows.

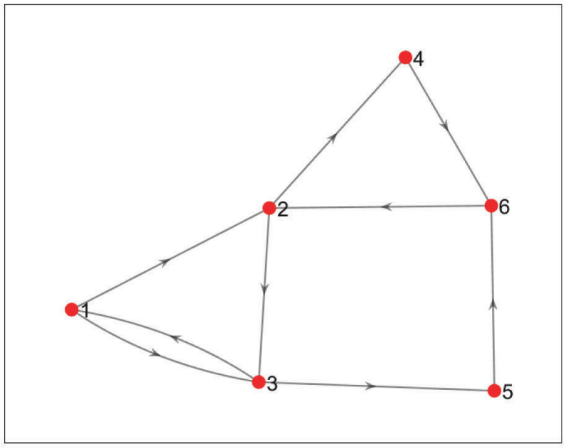


Figure 1: Directed Graph showing the communication links among the agents

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{1}$$

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix} \tag{2}$$

$$L = \begin{bmatrix} -1 & 2 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ -1 & -1 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix} \tag{3}$$

The weight of the edges is considered as 1 for easy interpretation. The adjacency matrix, degree matrix, and laplacian matrix are shown in Eqs. (1)-(3), respectively.

2.2. Consensus on Graph

Consensus of multiple agents on a graph can be achieved by a distributed control protocol. The important conditions [28] associated with the graph are given as follows.

Lemma 1. *[29, 30]: L has rank $N-1$, i.e., $\lambda_1 = 0$ is nonrepeated, if and only if graph G has a spanning tree.*

Lemma 2. *[28]: The local voting protocol guarantees consensus of single-integrator dynamics if and only if the graph has a spanning tree. Then, all node states come to the same steady-state values* $x_i = x_j, \forall i, j$.

2.2.1. Consensus for Single-Integrator System

Distributed control [28] for a Single integrator dynamic system is given as follows. The single integrator dynamics can be given by

$$\dot{x_i} = u_i \tag{4}$$

where x and u denote the state and the control, respectively. Distributed Control protocol is given by

$$u_i = \sum a_{ij}(x_j - x_i) \tag{5}$$

Therefore the dynamics is written as

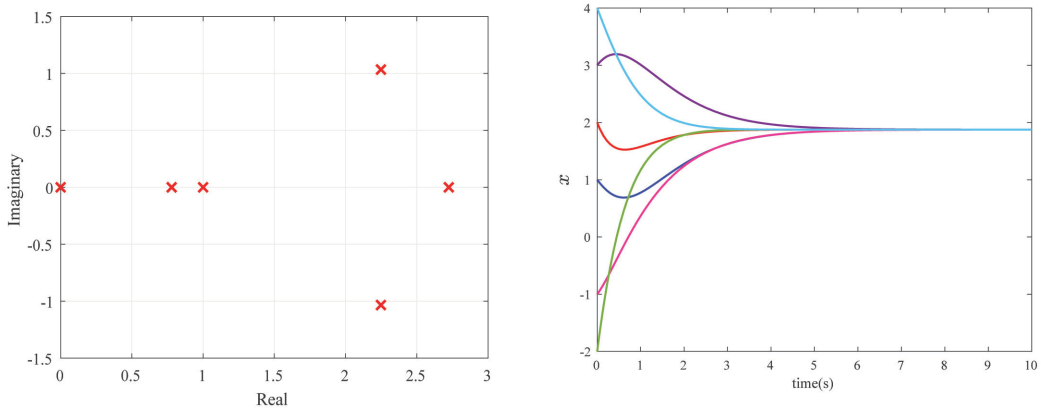
$$\begin{aligned} \dot{x_i} &= \sum a_{ij}(x_j - x_i) && (6) \\ &= -x_i \sum_{j \in N_i} a_{ij} + \sum_{j \in N_i} a_{ij} x_j \\ &= -d_i x_i + [a_{i1} \dots a_{iN}] \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} && (7) \end{aligned}$$

The global state $x = [x_1 \dots x_N]^T$ is given by

$$\dot{x} = -Dx + Ax = -Lx \tag{8}$$

The Distributed voting protocol is applied to the single integrator agents on the graph shown in fig. 1. The eigen values are shown in fig. 2a.

It can be observed that there is no repeated eigenvalue at zero. Therefore the graph has a spanning tree. According to Theorem 2, the voting protocol builds consensus among the agents on a graph. Figure 2b shows the agents with



(a) Eigenvalues of Laplacian

(b) Consensus of Agents

Figure 2: Consensus of Single-Integrator Agents

different initial conditions, attain a common value at the same time. Hence, Theorem 1, and 2 lead to achieving consensus among Multi-Agent Systems.

2.2.2. Consensus for Linear System

The consensus of agents with linear dynamics [28] is also discussed in this paper. The agent dynamics is considered as follows

$$\begin{aligned} \dot{Z}_i &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} Z_i + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \\ &= AZ_i + Bu \end{aligned} \tag{9}$$

where $Zi = [x_{1i}\ x_{2i}]$. The control expression for i^{th} agent is given as

$$u_i = K_i \sum_{j \in N_i} a_{ij}(Z_i - Z_j) \tag{10}$$

The gain of the feedback control law can be calculated using the LQR by solving ARE given as

$$A^T P + PA + Q - PBR^{-1}B^T P = 0 \tag{11}$$

where P is the solution of the ARE, $Q \geq 0, R > 0$. The expression of the gain obtained as follows

$$K_i = R^{-1}B^T P \tag{12}$$

3. Problem Formulation

It can be observed in fig. 2a that one eigenvalue (λ_1) is zero and others have a real positive part. Therefore the graph in fig. 1 has a spanning tree. It can be noted that the graph has more than one spanning tree, and the consensus can be built in many ways. Moreover, a number of such graphs are possible, which can have spanning trees. For each possible graph topology, there exists a different set of the neighbourhood and the agents require different control energy to achieve consensus on each of these network topologies. Therefore there is a scope to minimize the control energy require for consensus and find the network topology corresponding to the minimum energy (it is known as Optimal Topology). Therefore the problem can be given as

$$\min \sum_{i \in N_i} u_i^T u_i$$

The solution to this problem is the minimum energy and a network topology (specifically an adjacency matrix) which needs to satisfy constraints according to Theorem 1 and 2.

4. Two-Dimensional Genetic Algorithm (2D GA)

Generally, the GA works by the evolution of the chromosomes through generations. The chromosomes are usually a string of bits (0 and 1) or one-dimensional array of bits. But in this problem, the chromosome can't be represented by a one-dimensional bit string. Because in this case, the chromosomes are the adjacency matrix A. The two-dimensional chromosome representation is discussed in the following section.

4.1. Two-dimensional Chromosome representation

Two dimensional GA has been used to solve various problems. It is used in packing problem [31] which aims to obtain high packing density. 2D GA is

used for flight scheduling problem in [32] where the problem of scheduling of aircrafts is solved using 2D GA. The time table or schedule is considered as a 2D chromosome.

The variables required to describe a 2D chromosome is given in Table 1.

Table 1: Description of Variables

Variable	Definition
N	no of chromosome in the population
C_k	k^{th} chromosome, $1 \leq k \leq N$
$C_k(i,j)$	Gene at the position (i,j) in chromosome matrix
R	No. of rows of chromosome matrix
Q	No. of rows of chromosome matrix

The dimension of the chromosome matrix is considered as $R \times Q$. Therefore, rows and columns are denoted by $1 \leq i \leq R$, and $1 \leq j \leq Q$ respectively. An example of two-dimensional chromosome is given by

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{13}$$

A is an adjacency matrix which represents a directed graph as shown before. Each gene or each element of A represents the existence of a weighted directed edge, i.e., the status of the connection between any two specific agents. It can be noted that the weights are considered as 1 for simplicity.

4.2. Population Generation

In this work, the chromosomes are represented as square matrices because the adjacency matrix is square. It is important to note that the adjacency matrix

is not symmetric since it represents a directed graph. Also, the diagonal elements are zero. The algorithm to generate k^{th} chromosome C_k of the population of size N is given in Algorithm 1.

Algorithm 1 Initial Population generation

```
for i = 1 to R do
  for j = 1 to Q do
    x ← random number x ∈ (0, 1)
    if x > 0.5 then
      C_k(i, j) ← 1
    else
      C_k(i, j) ← 0
    end if
    if i = j then
      C_k(i, j) ← 0
    end if
  end for
end for
```

In the algorithm, the genes of a chromosome are created in a random manner. The genes at position (i, j) of the chromosome matrix, C_k is selected depending on a random variable $x \in (0, 1)$. If $x > 0$ then the value is selected as 1, otherwise 0. The diagonal elements are set to zeros. These chromosomes, thus generated, are provided as the initial population of the Genetic Algorithm.

4.3. Crossover

There are a few Crossover methods exist in the literature. Some of these methods are Multipoint Crossover [33], Uniform Crossover [34], One-Point Crossover [35], Substring Crossover [35]. More crossover methods can be found in [36].

Crossover method mentioned in [32], is adopted in this work. These methods are presented in algorithmic form. The 2D parent chromosomes are denoted by

'Parent 1', and 'Parent 2'. They are shown in fig. 3. The produced children are denoted by 'Child 1' and 'Child 2'.

Parent 1

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

$$\begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$

Parent 2

Figure 3: Parent Chromosomes

Algorithm 2 Substring Crossover

$r_1 \leftarrow$ random integer $< R$

$r_2 \leftarrow$ random integer $< Q$

$x \leftarrow$ random number $x \in (0, 1)$

if $x > 0.5$ **then**

Execute $Horizontal\_Crossover(r_1, r_2)$

else

Execute $Vertical\_Crossover(r_1, r_2)$

end if

The crossover point is selected in a random manner. Two random integers (r_1 and r_2) are generated, which are less than the maximum number of rows (R) and columns (Q) as given in Algorithm 2. An example of parent chromosomes is shown in fig. 3. The genes of Parent 1 is represented as a_{11} to a_{44}. Similarly, the genes of Parent 2 are represented by b_{11} to b_{44}. The crossover point is the gene at position (r_1, r_2) of parent chromosome matrices. The algorithm is

explained with the help of an example. In this example, the dimension of parent chromosome matrices is 4×4, i.e., $R = 4$, and $Q = 4$. The crossover position is obtained as $r_1 = 2$, $r_2 = 2$. Therefore the points of crossover for Parent 1 and Parent 2 are a_{22}, and b_{22}, respectively. Next, the type of crossover needs to be selected. For this purpose, a random variable x is considered, which can take any value between 0 and 1. As described in the algorithm, if the value of x is greater than 0.5, the horizontal crossover is selected. Otherwise, for $x < 0.5$, the vertical crossover is chosen. The horizontal crossover function $Horizontal\_Crossover()$ is described as follows. The rows or part of rows are exchanged between the parents.

Algorithm 3 function $Horizontal\_Crossover$

$Block1_{P1} \leftarrow Parent1(r_1, r_2 + 1 : Q)$

$Block2_{P1} \leftarrow Parent1(r_1 + 1 : R, 1 : Q)$

$Block1_{P2} \leftarrow Parent2(r_1, r_2 + 1 : Q)$

$Block2_{P2} \leftarrow Parent2(r_1 : R, 1 : Q)$

$Block1_{P1} \rightleftharpoons Block1_{P2}$ and $Block2_{P1} \rightleftharpoons Block2_{P2}$

The pictorial representation of Algorithm 3 is shown in fig. 4. According to the algorithm, the selected genes of Parent 1, i.e., a_{23} to a_{44} (shown in the red box) are replaced by selected genes of Parent 2, i.e., b_{23} to b_{44} (shown in the green box) to obtain Child 1. Similarly, a_{23} to a_{44} of Parent 1 is copied in place of b_{23} to b_{44} of Parent 2 to obtain Child 2. In this algorithm, a_{23} and a_{24} of Parent 1 are denoted by $Block1_{P1}$. The genes a_{31} to a_{44}, i.e., the third and fourth rows are denoted as $Block2_{P1}$. Similar elements of the Parent 2 are denoted as $Block1_{P2}$ and $Block2_{P2}$.

The Vertical Crossover algorithm is given in Algorithm 4.

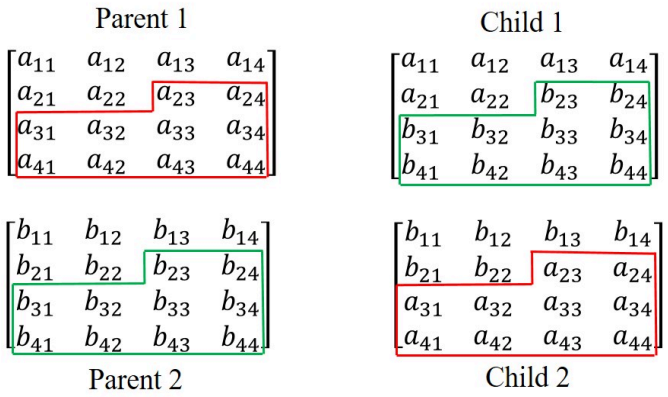


Figure 4: Horizontal Crossover: The selected rows and part of rows of Parent 1 and Parent 2 are exchanged.

Algorithm 4 function $Vertical\_Crossover$

$Block1_{P1} \leftarrow Parent1(r_1 + 1 : R, r_2)$

$Block2_{P1} \leftarrow Parent1(1 : R, r_2 + 1 : Q)$

$Block1_{P2} \leftarrow Parent2(r_1 + 1 : R, r_2)$

$Block2_{P2} \leftarrow Parent2(1 : R, r_2 + 1 : Q)$

$Block1_{P1} \rightleftharpoons Block1_{P2}$ and $Block2_{P1} \rightleftharpoons Block2_{P2}$

Vertical crossover is shown in fig. 5. In this case, genes of Parent 2, i.e., b_{32} to b_{44} (shown in the red box) are copied to the same positions of Parent 1, i.e., a_{32} to a_{44} (shown in the green box) to obtain Child 1. A similar operation is performed to obtain Child 2. In this case, a_{32} and a_{42} of Parent 1 are denoted by $Block1_{P1}$. The genes a_{13} to a_{44}, i.e., the third and fourth columns are denoted as $Block2_{P1}$. Similar elements of the Parent 2 are denoted as $Block1_{P2}$ and $Block2_{P2}$.

Another type of substring crossover is given in Algorithm 5. This is simplified version of horizontal and vertical crossover.

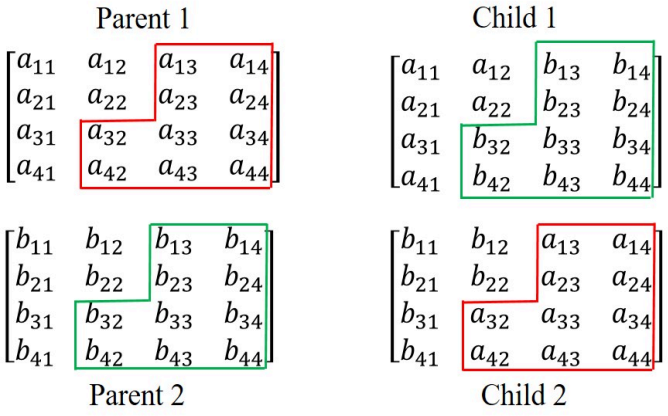


Figure 5: Vertical Crossover: The selected columns and part of columns of Parent 1 and Parent 2 are exchanged.

Algorithm 5 Substring Crossover

$r_1 \leftarrow$ random integer $< R$

$r_2 \leftarrow$ random integer $< Q$

Block1 $\leftarrow Parent1(r_1 : R, r_2 : Q)$

Block2 $\leftarrow Parent2(r_1 : R, r_2 : Q)$

Block1 $\rightleftharpoons$ Block2

According to this algorithm, the two random integers r_1 and r_2 are obtained. The block of genes in rows $r_1, r_1 + 1, \ldots, R$ and columns $r_2, r_2 + 1, \ldots, Q$ are interchanged between the two parents. The algorithm is explained with the help of an example, as shown in fig. 6. The block of genes are marked in the red box for Parent 1 and green for Parent 2. These genes are interchanged to obtain Child 1 and Child 2.

4.4. Mutation

The Mutation is an important operation to preserve the genetic diversity of a population of chromosomes in every generation. The Mutation is performed by exchanging one or more genes of the chromosomes. Generally, a certain percentage of the population is allowed to undergo mutation. The Mutation

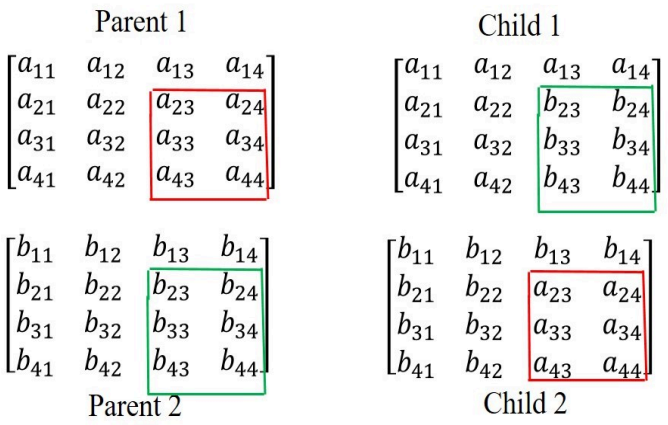


Figure 6: Substring Crossover: Selected block of genes are exchanged between the parents.

may change the solution considerably from the previous solution. Hence GA can come to a better solution by using mutation. Few mutation operations are shown in the following algorithms.

4.4.1. Two-Dimensional String Swapping Mutation

The process for this mutation is given in Algorithm 6.

Algorithm 6 String Swapping Mutation

$x \leftarrow$ random number $x \in (0, 1)$
if $x > 0.5$ **then**
  Execute $Horizontal\_Swap()$
else
  Execute $Vertical\_Swap()$
end if

The selection of the mutation type is purely random. It depends on a random variable $x \in (0, 1)$. If the value of x is greater than 0.5, then Horizontal Swap function, i.e., $Horizontal\_Swap()$ is executed. Otherwise, $Vertical\_Swap()$ is executed.

Algorithm 7 *Horizontal_Swap*()

$m_1 \leftarrow$ random integer $< R$

$m_2 \leftarrow$ random integer $< R$

if $m_1 \neq m_2$ **then**

Swap m_1^{th} and m_2^{th} rows of C_k

end if

Horizontal_Swap() function is given in Algorithm 7. It swaps m_1^{th} and m_2^{th} rows of a chromosome. The pictorial representation of the operation is shown in fig. 7. Let us consider, $m_1 = 1$, and $m_2 = 3$. Therefore, the first and third rows are swapped, as shown in the figure.

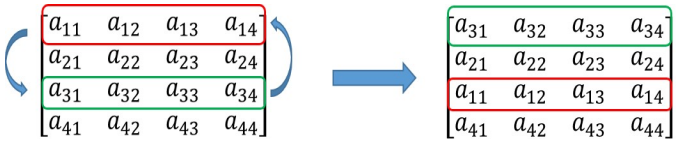


Figure 7: Horizontal Swapping: The selected rows are shown in red and green box. They are swapped.

Vertical_Swap() function is given in Algorithm 8. It swaps m_1^{th} and m_2^{th} columns of a chromosome. The pictorial representation of the operation is shown in fig. 8. Let us consider, $m_1 = 2$, and $m_2 = 4$. Therefore, the second and fourth columns are swapped, as shown in fig. 8.

Algorithm 8 *Verical_Swap*()

$m_1 \leftarrow$ random integer $< Q$

$m_2 \leftarrow$ random integer $< Q$

if $m_1 \neq m_2$ **then**

Swap m_1^{th} and m_2^{th} columns of C_k

end if

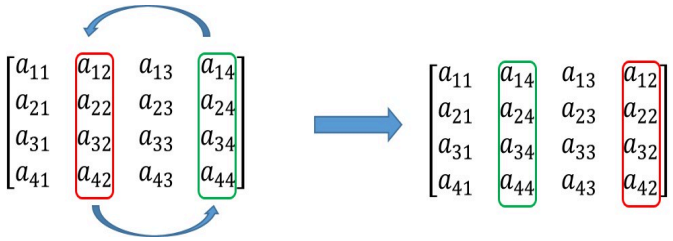


Figure 8: Vertical Swapping: The selected columns are shown in red and green box. They are swapped.

4.4.2. Two-Dimensional Two-Point Swapping Mutation Operation

Another type of mutation process is described in this section. It is named as Two-point swapping. The algorithm is given as follows.

Algorithm 9 $Two - Point\_Swap()$

$m_1 \leftarrow$ random integer $< R$

$n_1 \leftarrow$ random integer $< Q$

$m_2 \leftarrow$ random integer $< R$

$n_2 \leftarrow$ random integer $< Q$

if $m_1 \neq m_2$ **and** $n_1 \neq n_2$ **then**

Swap $C_k(m_1, n_1)$ and $C_k(m_2, n_2)$

end if

In this algorithm, two pairs of random integers are generated viz., (m_1, n_1), and (m_2, n_2). Genes at these two positions are swapped. An example of this operation is shown in fig. 9. The two random pairs are generated as $(3, 3)$ and $(1, 4)$. These elements (shown in the red and green box) are swapped, as shown in the figure.

5. Results

In this section, the simulation results are presented. The simulation is performed to find the optimal topology for six agents.Two types of agent dynamics are considered i.e., single integrator and Second-order linear system.

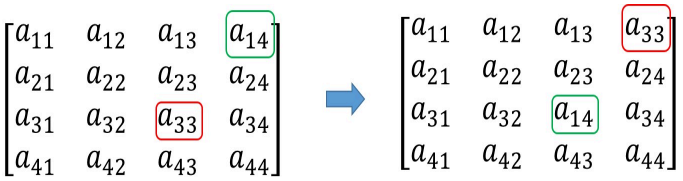


Figure 9: Two-Point Swapping: genes at two positions of a parent chromosome are swapped.

5.1. Optimal topology for Single Integrator System

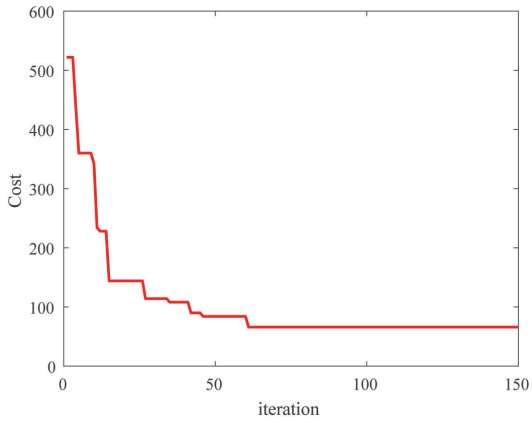
The initial conditions of the agents are considered as $x_0 = [1\ 2\ -3\ 4\ -2\ -1]$. The output of the two dimensional GA is an optimal adjacency matrix which is given in Eq. (14).

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}^T \tag{14}$$

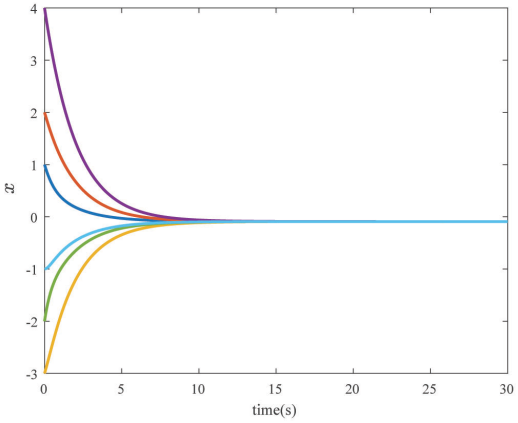
The graph corresponding to the optimal adjacency matrix is shown in fig. 10d. It can be observed that the directed graph shown in fig. 10d has a spanning tree. The existence of the spanning tree can be confirmed by the eigenvalues of the Laplacian matrix. These eigenvalues are shown in fig. 10c. It is clear that one of the eigenvalues is zero, and the rest of them have positive real part. Hence, there is no repeated eigenvalue at zero, which is an important requirement for a graph to contain a spanning tree.

The consensus among the agents on the optimal topology is shown in fig. 10b. All the agents with different initial conditions reached a common value within a few seconds on the graph.

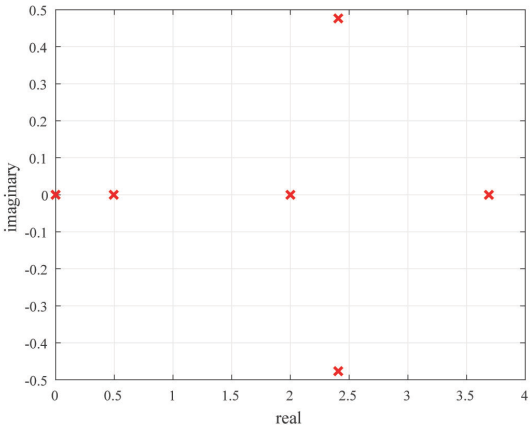
The cost or fitness value for various graphs are shown in fig. 10a. It can be observed that the cost decreases as the iteration of GA progress and finally, it reaches the lowest value 66 at 61^{st} iteration.



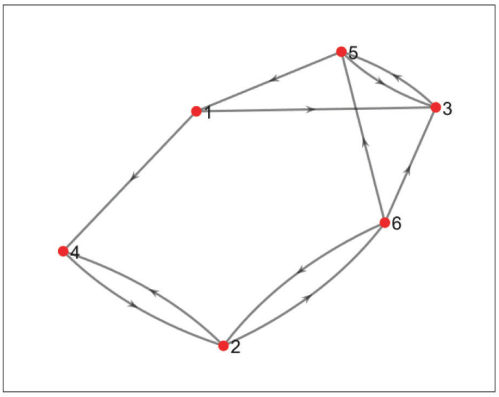
(a) Cost obtained using GA



(b) State trajectories of the Agents



(c) Eigen values of Graph Laplacian matrix



(d) Optimal Graph obtained by GA

Figure 10: Results obtained by two-dimensional GA

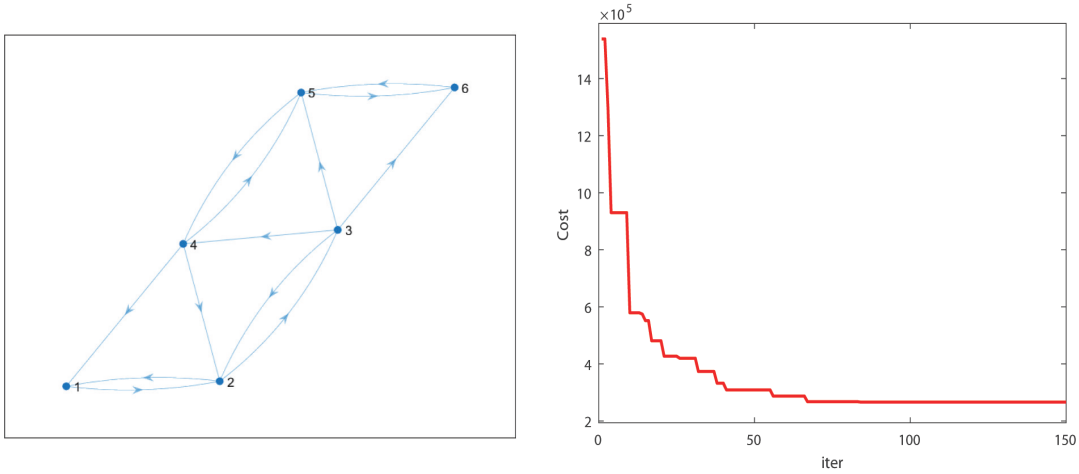
5.2. Optimal Topology for Linear system

The linear dynamics considered for the agents is discussed in section 2.2.2. The value of the gain matrix is considered as

$$K_i = \begin{bmatrix} -10 & 0 \\ 0 & -20 \end{bmatrix}$$

It can be observed that the control energy is minimized, as shown in fig. 11b. The optimal graph obtained is shown in fig. 11a. It can be observed that the

optimal graph obtained for linear dynamic agents is different from the single-integrator case. It is obvious because of the different dynamics, control expression, and initial conditions. The state trajectories (figs. 12a and 12b) show that the agents have achieved consensus within a few seconds. The control for the consensus is shown in fig. 13. Hence, the performance of two-dimensional GA for solving the optimal topology problem is satisfactory.



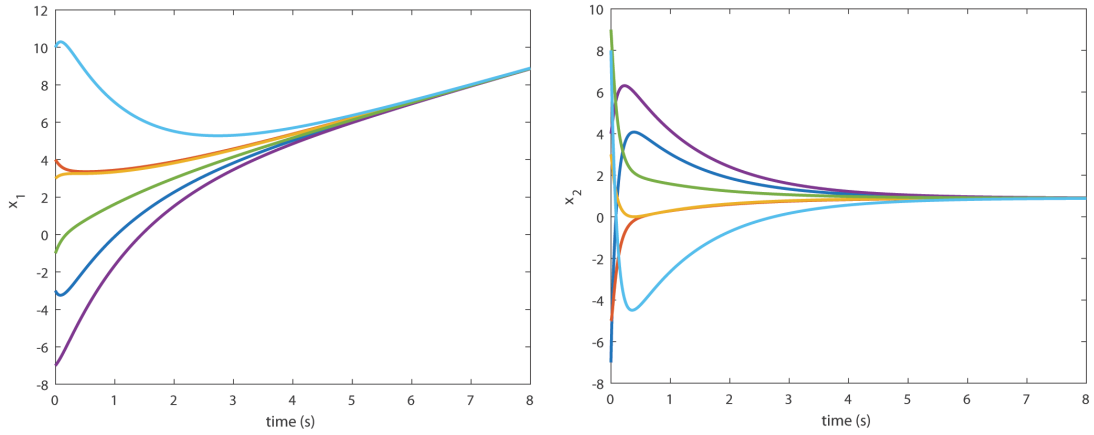
(a) Optimal Graph obtained using GA

(b) Cost minimized by GA

Figure 11: Performance of GA

6. Conclusion

This paper presents the bio-inspired approach to solve the optimal topology problem. It relaxes the need for direct communication between the leader and followers in a fixed star topology which was obtained by LQR presented in a few recent works. The results show that the agents have reached consensus by spending minimum control energy on the optimal communication topology obtained by two-dimensional GA, which is a novel approach. It is also possible to include more agents in the network and reconfigure the topology comfortably. It is because the inclusion of agents requires a change of dimension of adjacency matrix in the initial population, which is scope for future work. The technique



(a) State Trajectory x_1

(b) State trajectory x_2

Figure 12: State Trajectories

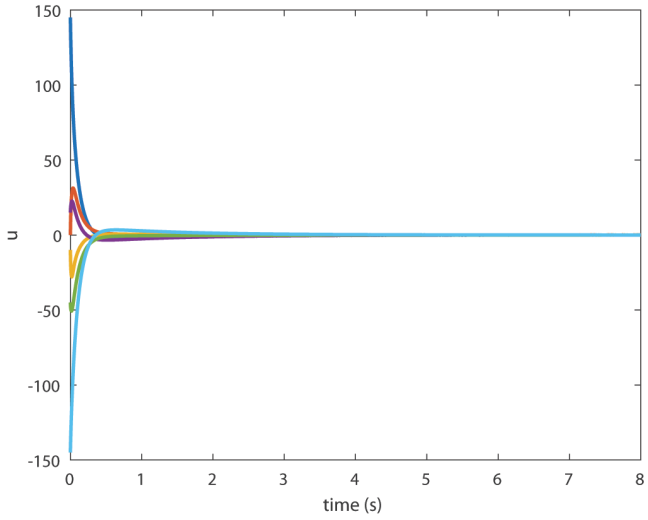


Figure 13: Control of Agents u

will help the user to find a specific communication topology which is more energy efficient towards achieving consensus.

Acknowledgment

This research was partially funded by an Engineering and Physical Sciences Research Council (EPSRC) project CASCADE (EP/R009953/1).

References

[1] W. Ren, R. W. Beard, Consensus seeking in multiagent systems under dynamically changing interaction topologies, IEEE Transactions on automatic control 50 (5) (2005) 655–661.

[2] R. Olfati-Saber, J. A. Fax, R. M. Murray, Consensus and cooperation in networked multi-agent systems, Proceedings of the IEEE 95 (1) (2007) 215–233.

[3] Y. Xie, Z. Lin, Global optimal consensus for multi-agent systems with bounded controls, Systems & Control Letters 102 (2017) 104–111.

[4] Y. Yin, Y. Shi, F. Liu, K. L. Teo, S. Wang, Second-order consensus for heterogeneous multi-agent systems with input constraints, Neurocomputing 351 (2019) 43–50.

[5] X. Liu, Z. Ji, T. Hou, H. Yu, Decentralized stabilizability and formation control of multi-agent systems with antagonistic interactions, ISA transactions 89 (2019) 58–66.

[6] T. Nguyen, H. M. La, T. D. Le, M. Jafari, Formation control and obstacle avoidance of multiple rectangular agents with limited communication ranges, IEEE Transactions on Control of Network Systems 4 (4) (2016) 680–691.

[7] J. Cao, G. Chen, P. Li, Global synchronization in an array of delayed neural networks with hybrid coupling, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 38 (2) (2008) 488–498.

[8] W. Yu, J. Cao, G. Chen, J. Lu, J. Han, W. Wei, Local synchronization of a complex network model, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39 (1) (2008) 230–241.

[9] A. Jadbabaie, J. Lin, A. S. Morse, Coordination of groups of mobile autonomous agents using nearest neighbor rules, Departmental Papers (ESE) (2003) 29.

[10] U. Munz, A. Papachristodoulou, F. Allgower, Consensus in multi-agent systems with coupling delays and switching topology, IEEE Transactions on Automatic Control 56 (12) (2011) 2976–2982.

[11] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, O. Shochet, Novel type of phase transition in a system of self-driven particles, Physical review letters 75 (6) (1995) 1226.

[12] W. Ren, R. W. Beard, Consensus algorithms for double-integrator dynamics, Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications (2008) 77–104.

[13] Y. Xie, Z. Lin, Global optimal consensus for higher-order multi-agent systems with bounded controls, Automatica 99 (2019) 301–307.

[14] Y. Zheng, Y. Zhu, L. Wang, Finite-time consensus of multiple second-order dynamic agents without velocity measurements, International Journal of Systems Science 45 (3) (2014) 579–588.

[15] Y. Liu, H. Min, S. Wang, Z. Liu, S. Liao, Distributed consensus of a class of networked heterogeneous multi-agent systems, Journal of the Franklin Institute 351 (3) (2014) 1700–1716.

[16] H. Wang, J. Ma, Optimal topology for consensus of heterogeneous multi-agent systems, Neurocomputing 177 (2016) 594–599.

[17] Y. Zheng, L. Wang, Consensus of heterogeneous multi-agent systems without velocity measurements, International Journal of Control 85 (7) (2012) 906–914.

[18] W. Yu, G. Chen, M. Cao, J. Kurths, Second-order consensus for multiagent systems with directed topologies and nonlinear dynamics, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 40 (3) (2009) 881–891.

[19] K. You, L. Xie, Network topology and communication data rate for consensusability of discrete-time multi-agent systems, IEEE Transactions on Automatic Control 56 (10) (2011) 2262–2275.

[20] M. Cao, A. S. Morse, B. D. Anderson, Reaching a consensus in a dynamically changing environment: A graphical approach, SIAM Journal on Control and Optimization 47 (2) (2008) 575–600.

[21] M. Fiedler, Algebraic connectivity of graphs, Czechoslovak mathematical journal 23 (2) (1973) 298–305.

[22] R. Olfati-Saber, R. M. Murray, Consensus problems in networks of agents with switching topology and time-delays, IEEE Transactions on automatic control 49 (9) (2004) 1520–1533.

[23] G. Chen, Y. Song, F. L. Lewis, Distributed fault-tolerant control of networked uncertain euler–lagrange systems under actuator faults, IEEE transactions on cybernetics 47 (7) (2016) 1706–1718.

[24] Y.-J. Pan, H. Werner, Z. Huang, M. Bartels, Distributed cooperative control of leader–follower multi-agent systems under packet dropouts for quadcopters, Systems & Control Letters 106 (2017) 47–57.

[25] Y. Cao, W. Ren, Optimal linear-consensus algorithms: An lqr perspective, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 40 (3) (2009) 819–830.

[26] J. Ma, Y. Zheng, L. Wang, Lqr-based optimal topology of leader-following consensus, International Journal of Robust and Nonlinear Control 25 (17) (2015) 3404–3421.

[27] J. Ma, Y. Zheng, B. Wu, L. Wang, Equilibrium topology of multi-agent systems with two leaders: a zero-sum game perspective, Automatica 73 (2016) 200–206.

[28] F. L. Lewis, H. Zhang, K. Hengster-Movric, A. Das, Cooperative control of multi-agent systems: optimal and adaptive design approaches, Springer Science & Business Media, 2013.

[29] Z. Qu, Cooperative control of dynamical systems: applications to autonomous vehicles, Springer Science & Business Media, 2009.

[30] W. Ren, R. W. Beard, Distributed consensus in multi-vehicle cooperative control, Springer, 2008.

[31] S. Jain, H. C. Gea, Two-dimensional packing problems using genetic algorithms, Engineering with Computers 14 (3) (1998) 206–213.

[32] M.-W. Tsai, T.-P. Hong, W.-T. Lin, A two-dimensional genetic algorithm and its application to aircraft scheduling problem, Mathematical Problems in Engineering 2015.

[33] K. A. De Jong, W. M. Spears, A formal analysis of the role of multi-point crossover in genetic algorithms, Annals of mathematics and Artificial intelligence 5 (1) (1992) 1–26.

[34] G. Syswerda, Uniform crossover in genetic algorithms, in: Proceedings of the third international conference on Genetic algorithms, Morgan Kaufmann Publishers, 1989, pp. 2–9.

[35] L. B. Booker, D. E. Goldberg, J. H. Holland, Classifier systems and genetic algorithms, Artificial intelligence 40 (1-3) (1989) 235–282.

[36] A. Umbarkar, P. Sheth, Crossover operators in genetic algorithms: A review., ICTACT journal on soft computing 6 (1).

2020-05-08

Optimal topology for consensus using genetic algorithm

Mondal, Sabyasachi

Elsevier

Mondal S, Tsourdos A. (2020) Optimal topology for consensus using genetic algorithm. Neurocomputing, Volume 404, September 2020, pp.1-49
https://doi.org/10.1016/j.neucom.2020.04.107